AGH
University of Science and Technology

Faculty of Electrical Engineering, Automatics, Computer Science and Biomedical Engineering

PROJECT REPORT

Magnetic Levitation System

Authors: Marcin Pilarski, Jakub Majcher, Daniel Sędłak

Tutor: dr inż. Andrzej Tutaj

Subject: Capstone Project 2

Kraków, May 2024

1. Laboratory stand's description

The system consists of an electromagnet affixed to a frame and a hollow metal sphere on a short pedestal. The goal is to keep the sphere in a set position relative to the electromagnet.

Control is achieved by applying appropriate voltage to the electromagnet, in order to balance the gravitational force, affecting the sphere. The configuration of the actual system is visible in Figure 1.

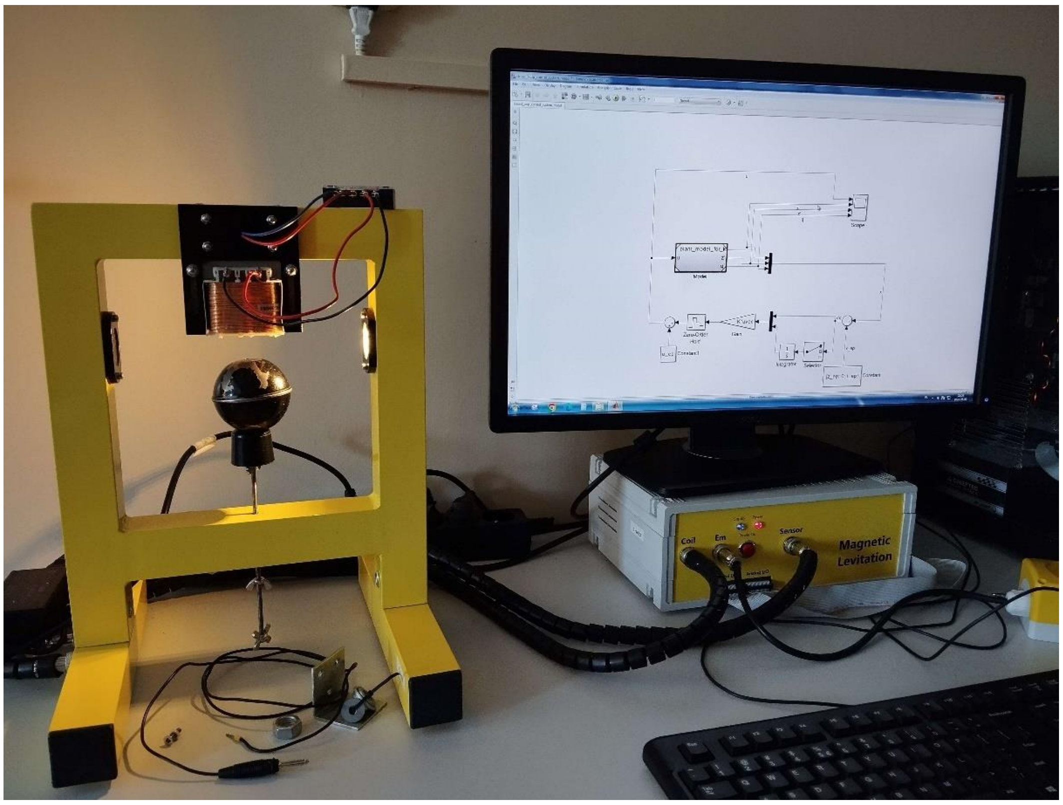


Figure 1. The laboratory magnetic levitation system

The MATLAB software had to be used to implement the control, along with the Simulink, Real Time Windows Target, and Real Time Workshop toolboxes. These toolboxes ensure that the signals exchanged between the PC and the electromagnetic system aren't delayed to the point of being unable to control the system at all. The input to the control system is a signal from a light detector – by measuring how much of the light emitted by the light source is obstructed by the sphere, it's possible to tell how far from the electromagnet it is.

2. Mathematical model of the object

In designing the mathematical description of the object, the convenient method of Euler-Lagrange equations was used. The mathematical model in Figure 2, describing the magnetic levitation system, was decided upon. First, the following model parameters were defined:

- z - the distance between the electromagnet and the surface of the levitating ball,
- q - load flowing through the winding of an electromagnet
- m - mass of the ball
- R - coil resistance
- g - standard gravity
- $L(z(t))$ - inductance of the coil, which depends on the position of the ball

The following energies act on the system:

- $\frac{1}{2}\mathrm{m}\dot{z}^2$ - kinetic energy of ball movement along the "z" axis of the system,
- $\frac{1}{2}\mathrm{L}\dot{q}^2$ - kinetic energy stored in a coil due to changes in the current flowing through
- $-mgz$ - gravitational potential energy
- $-uq$ - potential energy supplied to the system from an external source
- $\mathrm{R}\dot{q}^2$ - the energy emitted by the current flowing through the coil

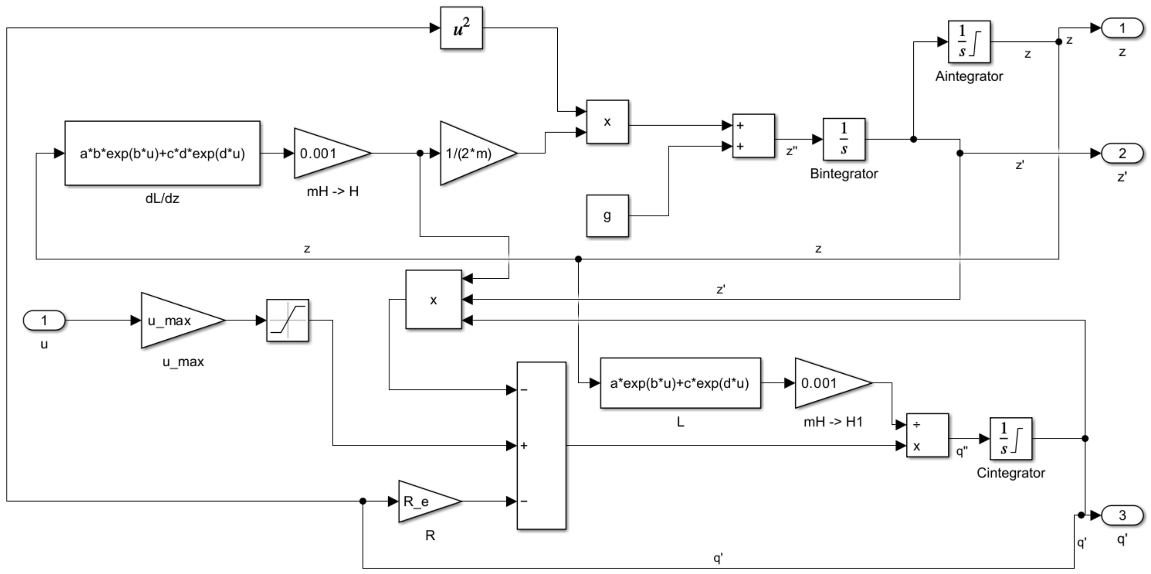


Figure 2. The mathematical nonlinear regulator object model in Simulink

3. Model's parameters' identification and calculations

In class, physical quantities and sensor signals were measured in order to identify the model. Measured or calculated parameters of the model are gathered in the following table:

Table 1. Parameters of magnetic levitation system

Parameter name	Description	Unit
$m = 0.058$	mass of the ball	$[kg]$
$g = 9.81$	gravitational acceleration	$\left[\frac{m}{s^2}\right]$
$R = 4.38$	coil resistance	$[\Omega]$
$u_{max} = 12.28$	max voltage	$[V]$

To be able to read measurements of the sensor, we had to create a plot of how the sensor's voltage signal corresponds to the distance in meters.

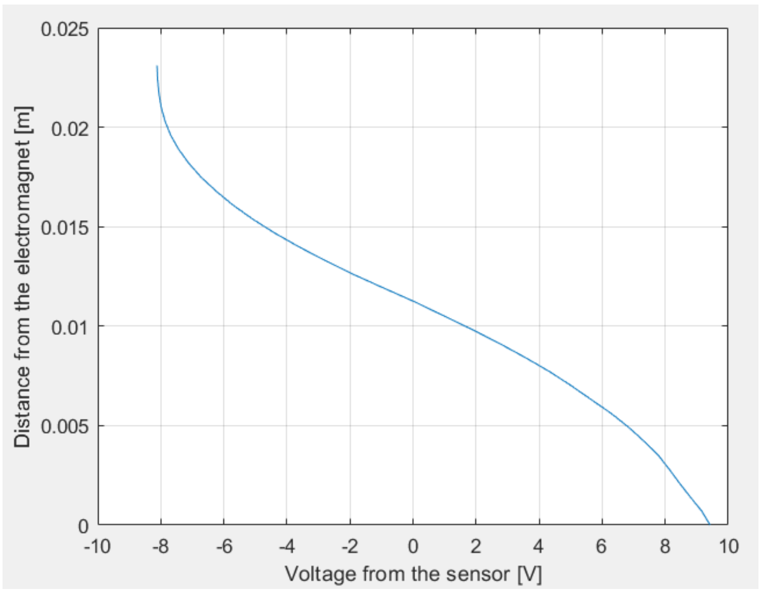


Figure 3. Voltage-distance profile of the object

Next on, we had to take measurements of the values of the inductive resistance, compared to the distance. In a later stage its formula, along with first and second order derivatives, will be used in the Lagrangian equations.
The resulting formula, obtained by fitting an exponential function, along with the derivatives, are:

$$L(z) = 27.62e^{-304.84z} + 112.85e^{-2.74z}$$

$$\frac{dL}{dz} = -8419.68e^{-304.84z} - 309.21e^{-2.74z}$$

$$\frac{d^2L}{dz^2} = 2566655.5e^{-304.84z} + 847.23e^{-2.74z}$$

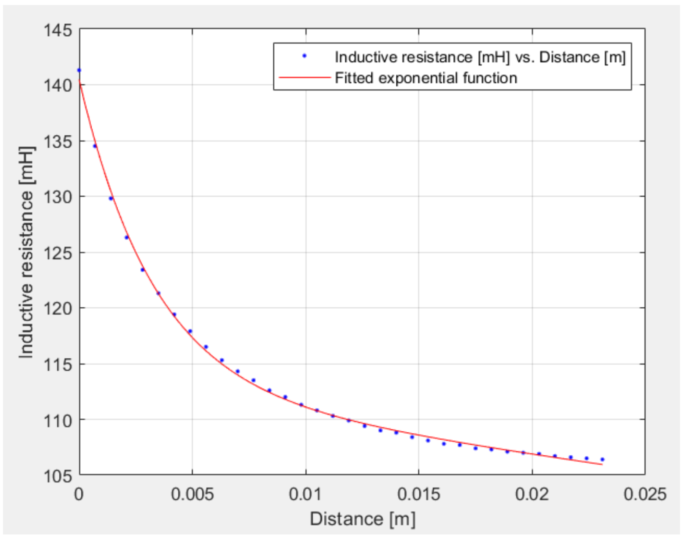


Figure 4. Inductive resistance and distance profile - measurements and a fitted exponential function

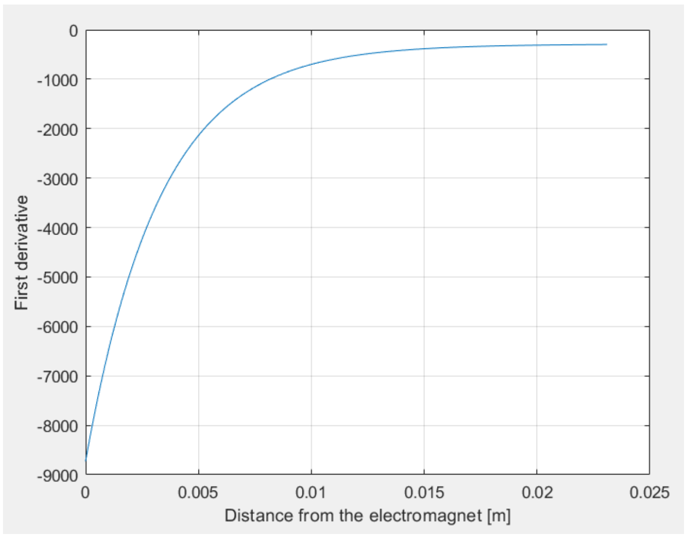


Figure 5. First order derivative of the inductive resistance and distance profile

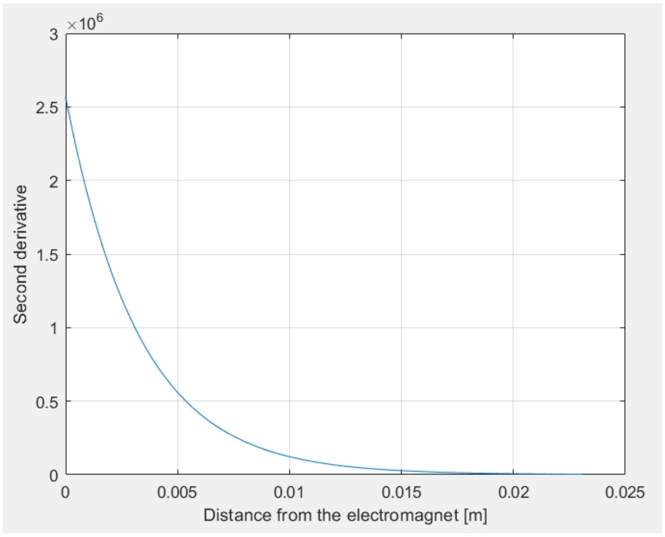


Figure 6. Second order derivative of the inductive resistance and distance profile

As a next step, we have measured a few values of the current and voltage in the electromagnet, in order to find a relationship between them. We then fitted a linear function which described this relationship – with parameters $a = 0.956$ and $b = 0.0401$. Using this linear function, we could now calculate the current in the program, using the voltage sensor's readings.

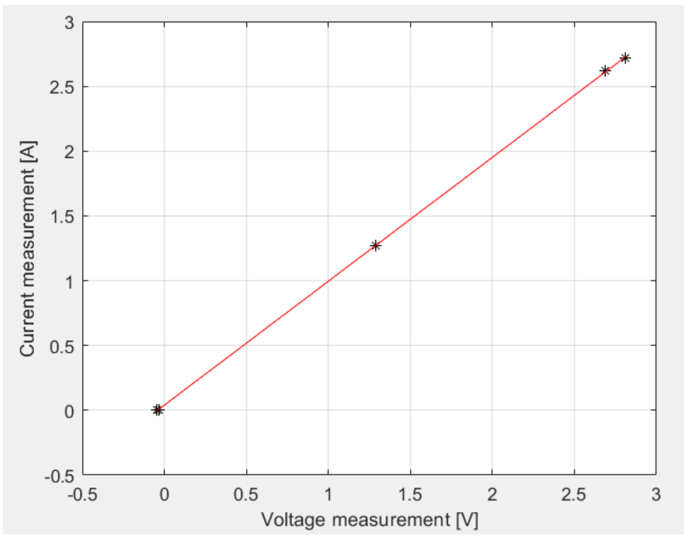


Figure 7. Measurements of the current and voltage values, along with a fitted line

We also compared the measurements of the current with the output voltage of the station – which gave us the value of resistance: $\mathrm{R} = 4.38\Omega$

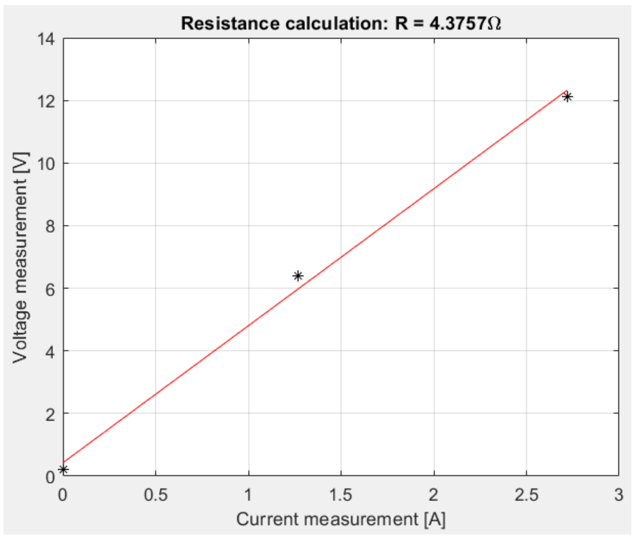


Figure 8. Measurements of the current and the output voltage of the station

Lastly, we compared the output voltage of the power station with the control values in the program, which are the duty factor of the PWM. These measurements allowed us to obtain the parameters of the linear function: $a = 11.924$ and $b = 0.356$, which in turn is needed to scale the control signal properly in the model.

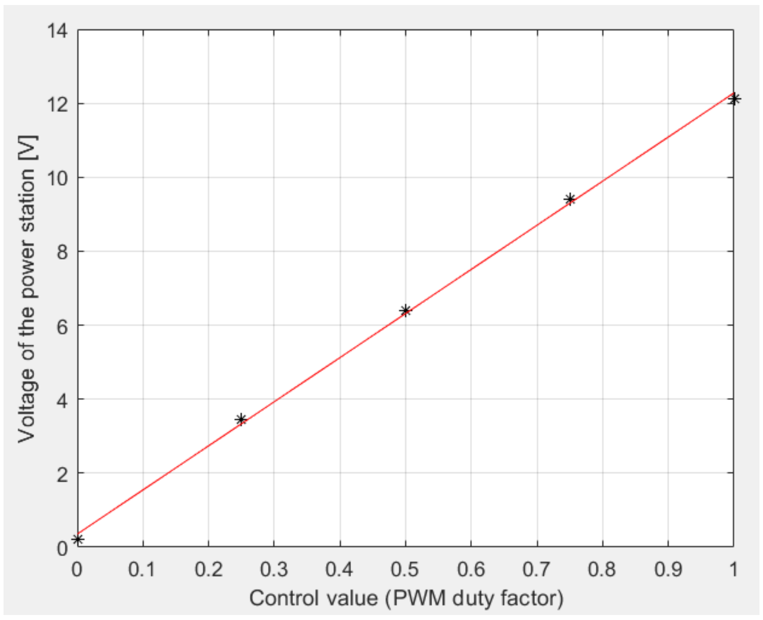


Figure 9. Linear relationship between the PWM duty factor and the output voltage

The completed model's response is very close to that of the object (Figure 10), except for the current – however we later noticed that the model was using a wrong variable for the resistance due to a collision of variable names. After the correction the current was reaching the same level as that of the object.

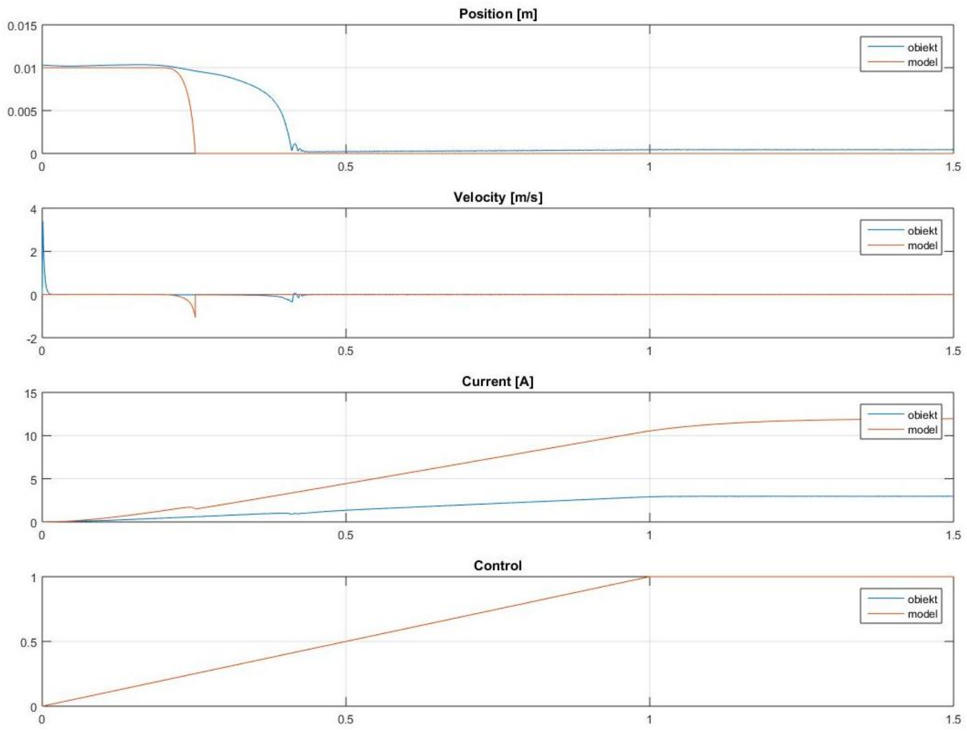
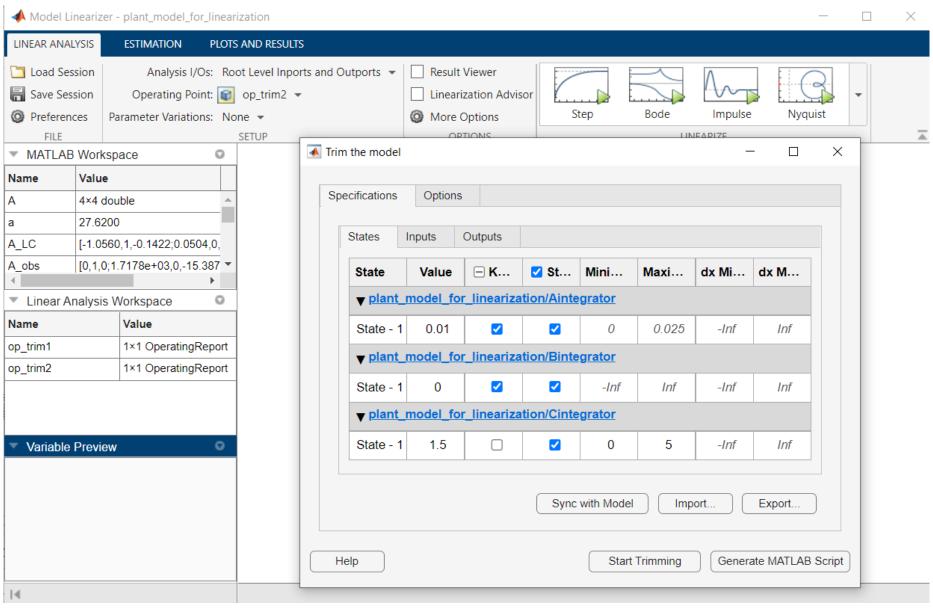


Figure 10. Comparison of the object's and the model's responses to the same input sequence

4. LQR regulator

In order to control the object and keep the ball in a set position, an LQR regulator has been designed. To this end, the model was linearized in an operating point – with a fixed position and velocity values. To calculate other appropriate values, like the differential of the states or the current, a special tool in MATLAB was used, called the Linear Analysis Tool (Figure 11). With its help, we were able to generate an operating point and linearize the model, obtaining its state matrices.



Analysis I/Os: Root Level Inports and Outports
Operating Point: op_trim2
Parameter Variations: None

MATLAB Workspace

Name	Value
A	4×4 double
a	27.6200
A_LC	[-1.0560,1,-0.1422;0.0504,0,
A_obs	[0,1,0;1.7178e+03,0,-15.387

Linear Analysis Workspace

Name	Value
op_trim1	1×1 OperatingReport
op_trim2	1×1 OperatingReport

State	Value	K...	✓ St...	Mini...	Maxi...	dx Mi...	dx M...
plant_model_for_linearization/Aintegrator							
State - 1	0.01	✓	✓	0	0.025	-Inf	Inf
plant_model_for_linearization/Bintegrator							
State - 1	0	✓	✓	-Inf	Inf	-Inf	Inf
plant_model_for_linearization/Cintegrator							
State - 1	1.5	☐	✓	0	5	-Inf	Inf

Figure 11. Creating an operating point in the Linear Analysis Tool

To test the resulting linear model, we checked the step response (Figure 12). As seen in the screenshot below, the response is running away into infinity, which is a sensible result – the object is unstable, after all.

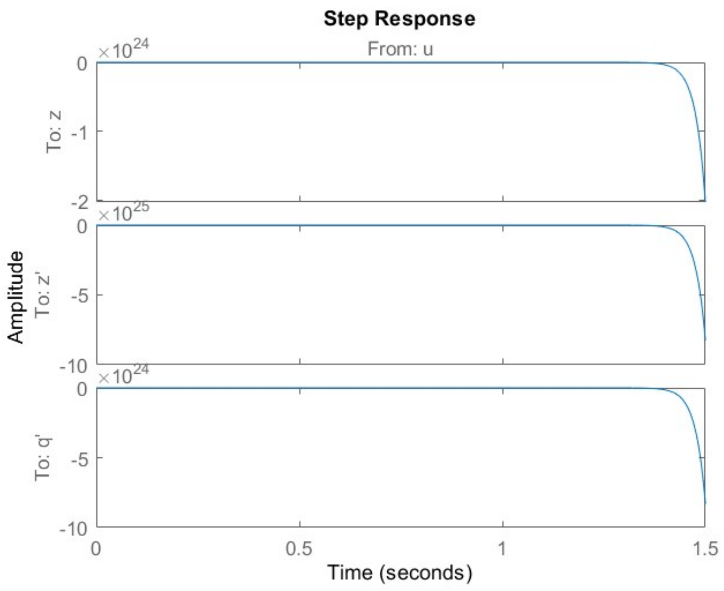


Figure 12. Step response of the linearized model

The next step is to design the actual regulator. We used the default (with values of 1) Q and R matrices, which in turn were used to compute the K matrix of the regulator's coefficients. After a bit of testing, we decided to change the values of the Q matrix, so some state variables are prioritized over others by the controller.

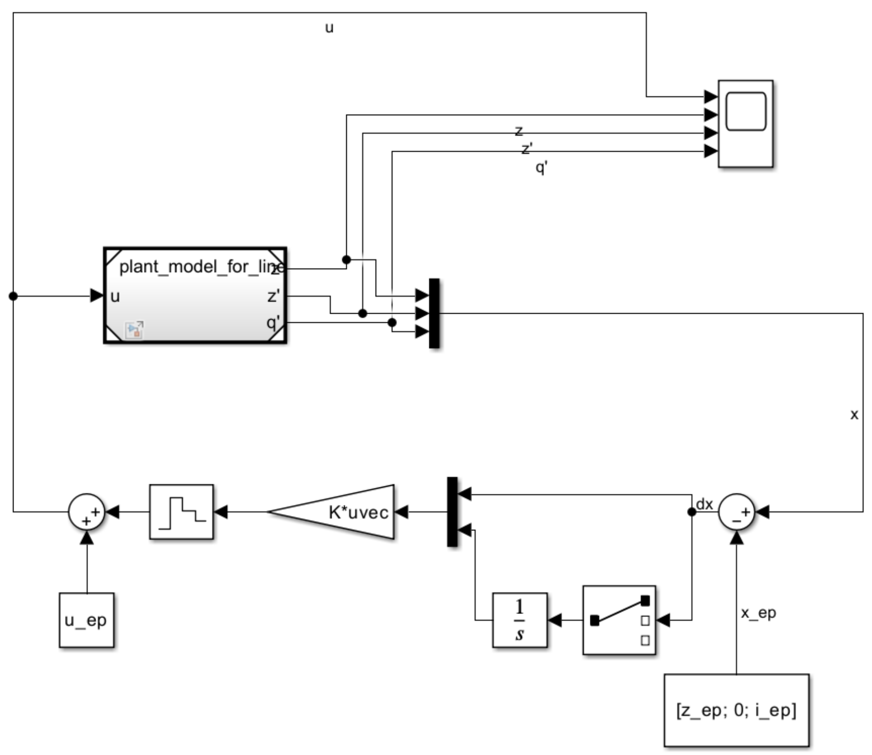


Figure 13. The implemented LQR controller

The results of adding the LQR controller can be seen in Figures 14 and 15.

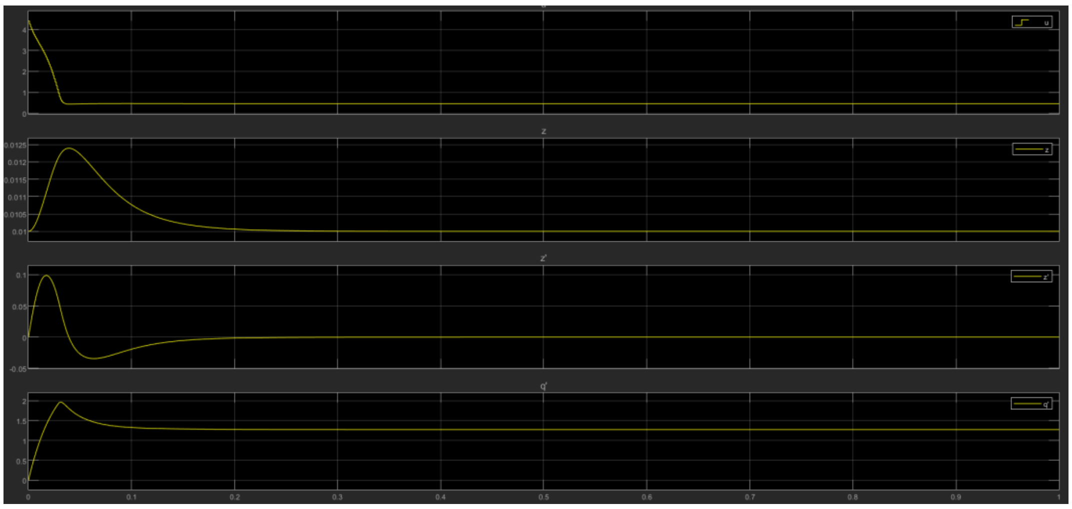


Figure 14. The model's behaviour with the added LQR controller

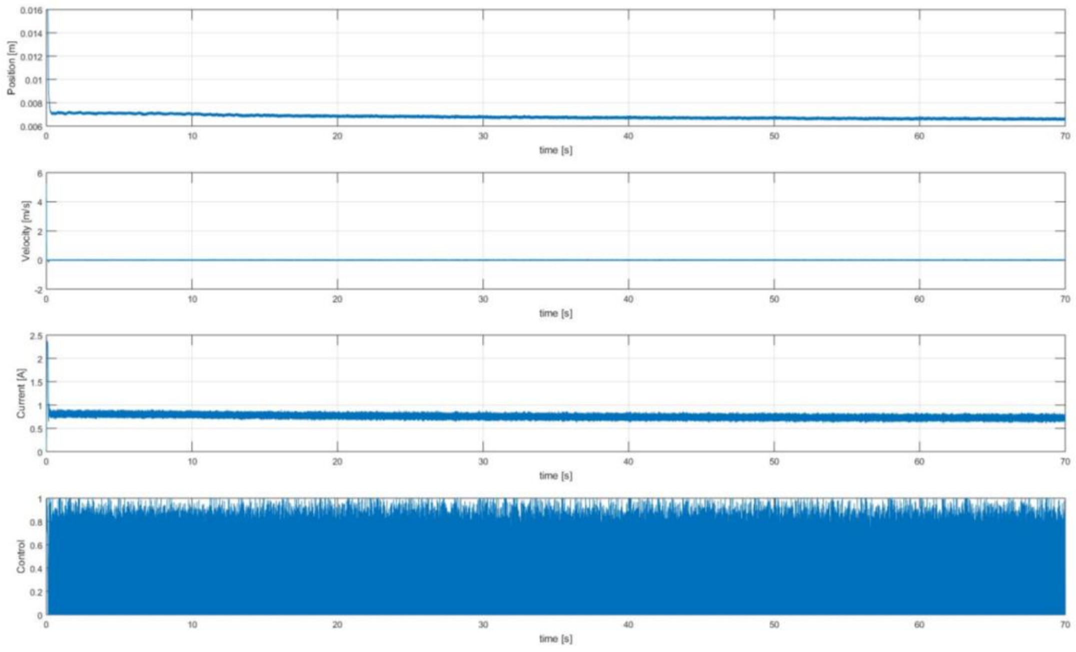


Figure 15. The object's behaviour with the added LQR controller

Unfortunately, as seen in Figure 15, there is a static error in the position of the ball, when the controller is used on the object. The deviation of the position adds to the deviation of the control signal, which falsely makes the controller see the situation as intended.

5. LQI controller

To get rid of the static error, we have added an integration part to the controller as the fourth state variable. The integrator was using the position error – because it is the most important variable to be stabilized. The resulting controller is an LQI controller.

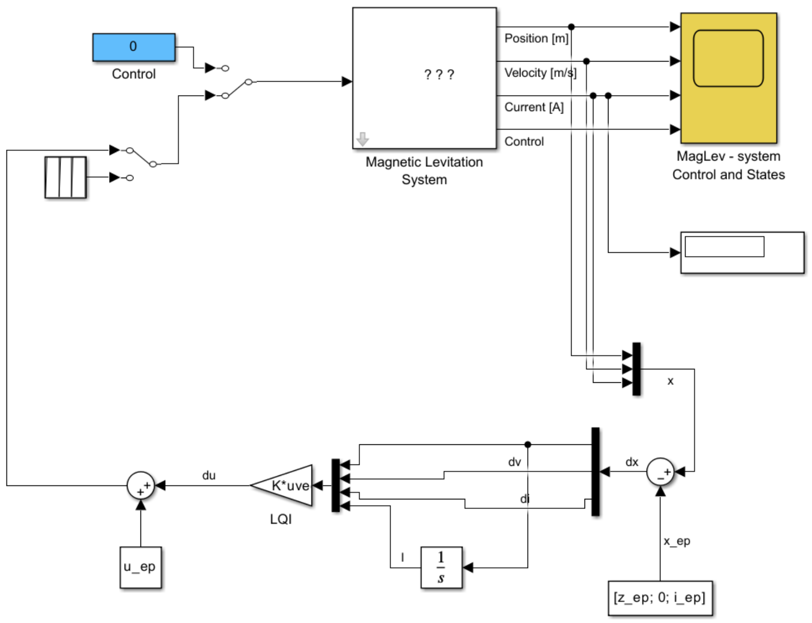


Figure 16. The Simulink driver for the controlled object with the LQI controller

There was no difference in how the model behaved with the new controller, as the previous version was already working perfectly fine – the problem in the behaviour of the real object was happening due to the discrepancy between the model and the reality. There was, however, a big improvement in the behaviour of the real object – as seen in Figure 17. The position reaches the near neighbourhood of the set position (0.01m) and then asymptotically rises further to align with it perfectly.

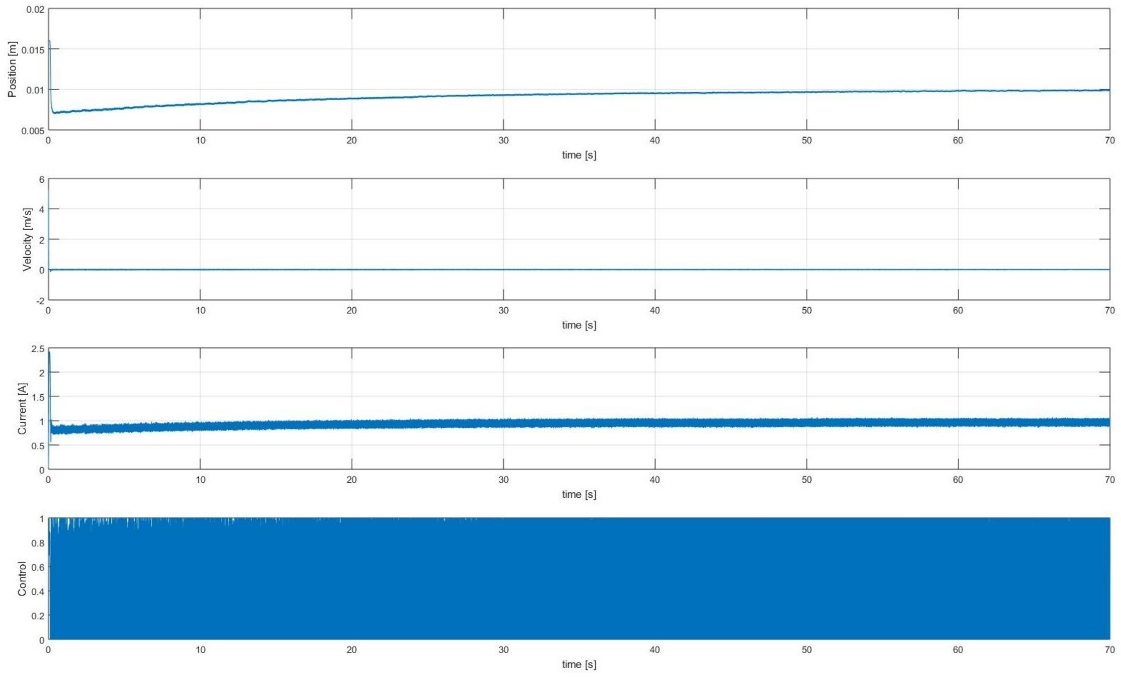
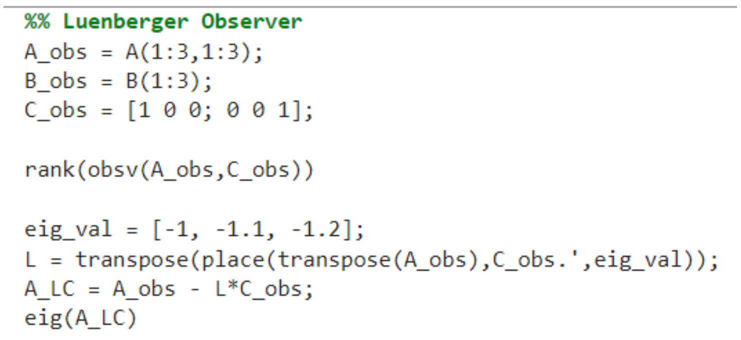


Figure 17. Behaviour of the object controlled by the LQI controller.

There is a problem with the new version of the control system however – the control signal is incredibly noisy, which likely means that it is being clipped at the limits (0 and 1).

6. Luenberger observer

One of the possible causes of a noisy control signal is a noisy state variable that forces the controller to change the control rapidly. Therefore, the best way to fix this issue is to remove the noise from the state variables, namely the speed and the current. For this reason, we have implemented the Luenberger observer – which will aim to provide state variable data without noise. It is trickier than the previous modifications we have implemented – the MATLAB code used for preparations is shown in Figure 18.



```
%% Luenberger Observer
A_obs = A(1:3,1:3);
B_obs = B(1:3);
C_obs = [1 0 0; 0 0 1];

rank(obsv(A_obs,C_obs))

eig_val = [-1, -1.1, -1.2];
L = transpose(place(transpose(A_obs),C_obs.',eig_val));
A_LC = A_obs - L*C_obs;
eig(A_LC)
```

Figure 18. MATLAB code preparing variables for the Luenberger observer's implementation in Simulink.

The A and B matrices are limited, so as not to count in the added integrated position error part. The C matrix had to be changed as well, so the speed is not used as the output variable – it is calculated by the observer after all. Next on, the rank of the resulting observability matrix is calculated to ensure the system is still observable after the changes. The command was returning the rank equal to the number of matrix's rows, which satisfies this condition. The error between the real and observed variables needs to asymptotically shrink to 0, which forces us to use eigenvalues with negative real values – they are later used in the calculation of the L matrix of the observer. Lastly, the eigenvalues of the $A - LC$ matrix are checked – but they are precisely equal to the chosen ones, so the preparation is completed successfully, and the Simulink model can be prepared (Figure 19).

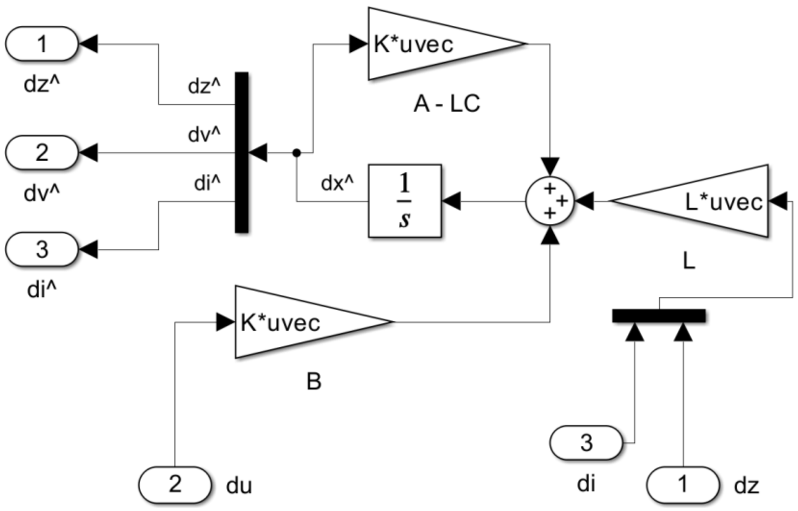


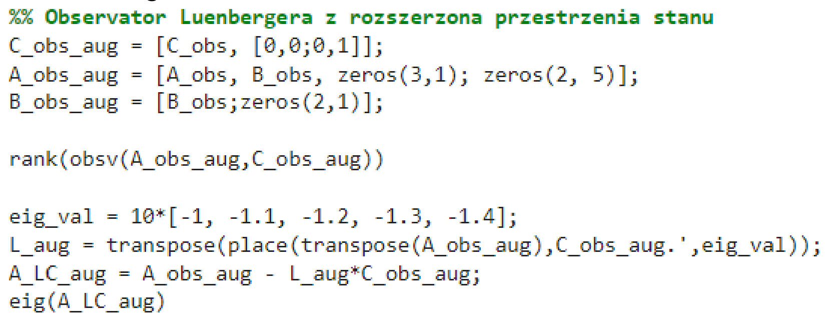
Figure 19. Simulink model of the Luenberger observer

The model implements the formula of the Luenberger observer:

$$\Delta\dot{\hat{x}} = A\Delta\hat{x} + B\Delta u + L(\Delta y - C\Delta\hat{x})$$

Test results of the observer implementation showed that Δz does not represent the actual state of the system with sufficient precision. An error of 1-2 cm was observed in the observed Δz, while the actual Δz was approximately 1 mm. This significant deviation indicates inaccuracy in determining the eigenvalues for the control signal and current.

Since the classic implementation of the observer did not bring the expected results, it was assumed that the observer is burdened with a constant error due to control and current. Then, the rank of matrices A, B and C was increased accordingly so that the observer in this implementation determined the error between the determined and actual eigenvalues of the current and control signal of the modelled system. The modified code needed for implementation is shown in Figure 20.



```
%% Observator Luenbergera z rozszerzona przestrzenia stanu
C_obs_aug = [C_obs, [0,0;0,1]];
A_obs_aug = [A_obs, B_obs, zeros(3,1); zeros(2, 5)];
B_obs_aug = [B_obs;zeros(2,1)];

rank(obsv(A_obs_aug,C_obs_aug))

eig_val = 10*[-1, -1.1, -1.2, -1.3, -1.4];
L_aug = transpose(place(transpose(A_obs_aug),C_obs_aug.',eig_val));
A_LC_aug = A_obs_aug - L_aug*C_obs_aug;
eig(A_LC_aug)
```

Figure 20. Augmented Luenberger Observer Matrices

Then, an experiment was carried out to compare the system's response to control using the previously determined speed and the speed determined by the observer. The implementation of the regulation in Simulink is visible in Figure 21.

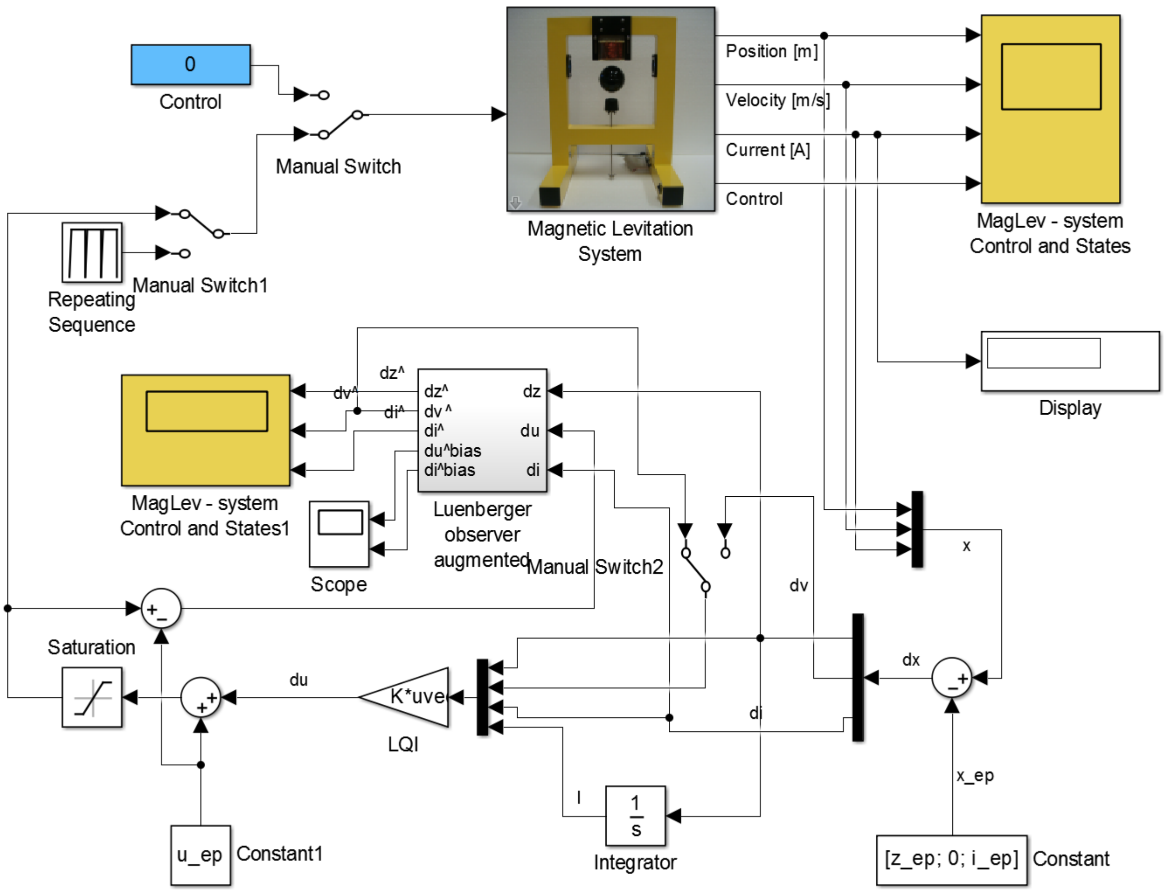


Figure 21. Simulink LQR with observer implementation.

Observer implementation results:

1. Improvement in reaching the set position

In the experiment, a significant improvement was observed in reaching the set position, which is 10 mm from the coil. The levitation system maintains the set position more effectively, which proves better stability and precision of the Luenberger observer.

2. Noise reduction in speed signal

Reducing noise in the speed signal was one of the key results of the experiment. Lower noise in this signal resulted in more stable and accurate speed measurements, which is important for precise system control.

3. Limiting the value of the control signal

By reducing noise in the speed signal, the value of the control signal is also limited. This means that the control signal did not exceed the minimum and maximum allowable values. In the context of a magnetic levitation system, this means that the PWM signal remained within operating limits, which prevented the control signal from going into saturation.

4. Elimination of the "clipping" effect

By keeping the control signal within the minimum and maximum values, the "clipping" effect was eliminated. When the control signal is saturated, the average signal value may differ significantly from the actual value needed to maintain system stability. Eliminating this effect allowed for more precise control and better compliance of the control signal with the actual system requirements.

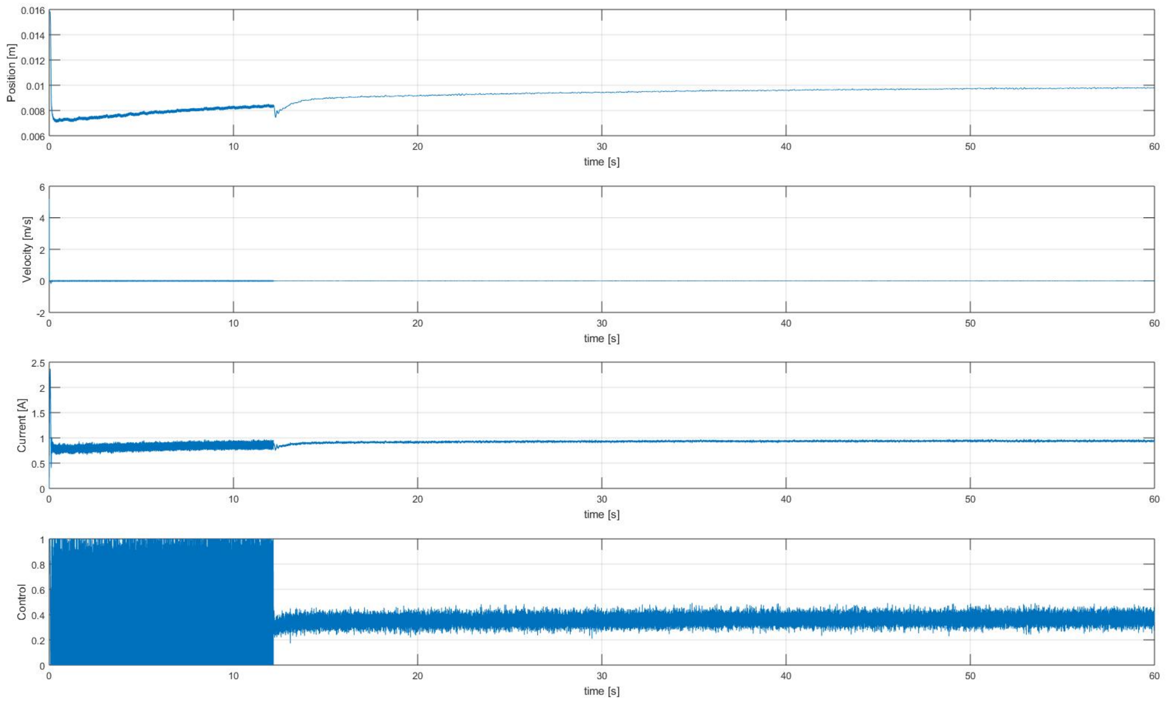


Figure 22. Observer implementation results

8. Summary

The project involved several tasks. Initially, a mathematical model based on the theoretical Euler-Lagrange equations was developed and implemented in Simulink. The system parameters were then determined using various measurements. The next step to control the object and keep the ball in a fixed position was to linearise the system to calculate the LQR controller parameters. These parameters were then tuned to obtain better results. Unfortunately, when the controller was applied to the object, a static error occurs in the ball's position. This position deviation was added to the control signal deviation, causing the controller to incorrectly interpret the situation as intended. To address this issue, an LQI controller was designed, incorporating an integrating component as a fourth variable. The model's behaviour remained unchanged with the new controller LQI, but the real object's performance significantly improved despite discrepancies between the model and reality. The last step was to implement a Luenberger observer, designed to provide noise-free state variable data. Based on the results of the experiments and the comparison of the values of the state variables from the object model with the calculated parameters, it can be concluded that the constructed observer Luenberger works correctly.

In the task of stabilising a metal ball at a preset distance of z = 0.01 m, despite the presence of various disturbances, the object achieved its preset position, also the additional appearance of an external force was adequately compensated.